A2-Central™

formerly
Open-Apple

August 1990
Vol.6, No.7

ISSN 0885-4017
newstand price: $2.50
photocopy charge per page: $0.15

A journal and exchange of Apple II discoveries

Apple speaks (and listens)

How visible is the Apple II community's dissatisfaction with Apple? Well, David Reed's report on the National Apple Users Group Conference in the July issue of *Byte* magazine ("What's New" Midwest regional supplement) mentioned the visibility of the "Apple II Infinitum" campaign at the Chicago meeting, and Apple II Product Manager Rajiv Mehta's attempts to confirm the viability of the Apple II during his speech at the event. Apple apparently feels the outcry of Apple II users reflects an image of the company that they cannot accept, and (as reported in "Miscellanea" last month) has assigned seven-year Apple veteran Ralph Russo a highly-placed position within Apple to evaluate and respond to the problem.

During June 15-17, Jim Merritt (leader of Apple's Apple II "Green Software" group responsible for applications, utilities, and development environments) attended Apple Fiesta, a regional Apple II and Macintosh conference hosted by the AZ-Apple user group in Phoenix, Arizona. Merritt was "between fires" at Apple and was asked by Ralph Russo to combine the trip to Apple Fiesta with a short vacation. He brought a notebook and spent much of his time wandering the conference and gathering comments and suggestions regarding Apple products, primarily the Apple II product line, as well as giving two talks regarding future directions at Apple.

Merritt portrayed the Russo organization as a "potent but small core" within Apple dedicated to cutting through Apple's bureaucratic net and finding ways of getting the most out of the Apple II product line. The group was described as being "small, but not too small"; large enough to get the job done, but not so large as to have bureaucratic problems of its own.

Apple is working on more hardware and software for the Apple II as well as the Macintosh. Among the primary technology directions Apple is concerned about are "media integration" (Merritt used this more descriptive term in place of "multimedia"), data exchange between the two dominant Apple product lines, and universal access regardless of the dominant sensory mode of the user.

Media integration consists of the ability to use the computer with other devices meant to convey information by various methods: hard disks, video monitors, CD-ROM drives, laser disc systems, and so on (see "Picture This", July 1990).

Data exchange will focus on the ability to move data files between the Apple II and Mac platforms so that integrating the two types of systems within an environment will not cause an unnecessary burden on the user. The audience voiced its concern that such exchanges be facilitated in *both* directions, unlike some past attempts (such as the third-party *Works-to-Works Transporter* utility for *Apple File Exchange*). Apple has managed to converge the line of peripherals for both systems over the past few years so that the Apple II can work with the same 3.5 drives, SCSI hard disks, printers, and so forth as the Macintosh, which will help in making the systems more compatible when combined in a single environment.

Universal access refers to the broadening of Apple's "Human Interface" design to allow computer users who may not favor (or be able to use efficiently) specific input/output methods to have alternatives available. One example is the *Apple IIgs Video Keyboard v.1.0B1* (currently a beta product available to APDA members for $20, part #A0028LL/A) a New Desk Accessory which provides an on-screen keyboard accessible via the mouse (or other compatible pointing device).

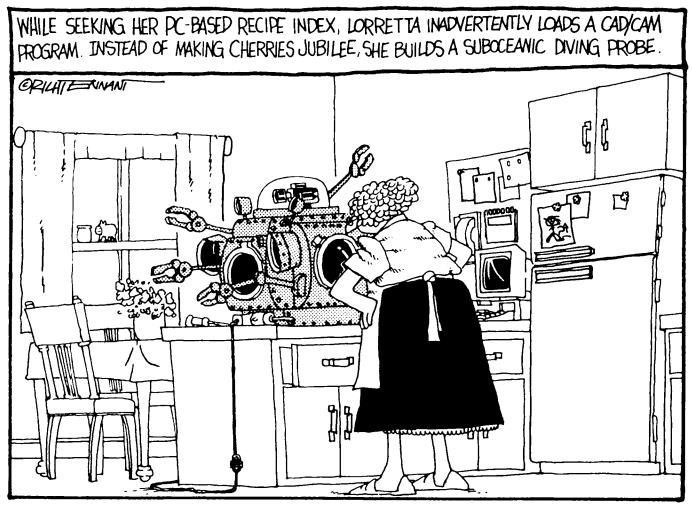
All of these directions seem to be logical choices for Apple if it is to solve the problem of integrating Macintosh and Apple II systems among its customers. And it is unnecessary to be paranoid about Apple's motives; you can read them by tallying the number of systems "in place" in Apple's important K-12 education market. Using rough figures based on a survey published in the October 1989 issue of *Electronic Learning*, of the 59% of the Apple computers in K-12 institutions (a figure close to the 60% figure Apple has used), 97% are Apple II systems. If Apple wants to sell Macs to this market, it had better demonstrate that it isn't declaring its current systems obsolete after only a few years use.

About the Apple IIgs specifically, Merritt stated that he felt its system software represented the best built-in "toolbox" for developers ("in my humble opinion") of any personal computer. Regarding the 8-bit Apple II systems, Apple is looking into ways to bring the desktop metaphor to those platforms.

Apple is also investigating the codification of their text (MouseText and file card) interfaces for Apple II systems so a more consistent interface (across applications) for text applications can be documented in the same manner as the desktop metaphor is for applications using the graphics display.

Merritt also affirmed that Apple is indeed looking into new Apple II CPUs, though he cautioned that the decision of bringing a new CPU to market depended on more than just technological issues.

In his opening remarks in one speech, Merritt emphasized that the company very much wants to get back the advantage that it had when people bought Apple computers partially because of a "grassroots" feeling that the company and its products were accessible. Apple wants the feeling of using Apple products to be a positive and enjoyable experience from the time the computer is taken from the box. Apple realizes many Apple II (and even Macintosh) users may not have felt this way in recent years. To help correct this, Apple wants to improve user group and developer relations, including an elimination of the schism that seems to exist between Macintosh and Apple II



factions. Also, extending John Sculley's comments about the work on lowering the cost of Macintosh systems, Apple is investigating ways to lower the cost of many of its products, including peripherals and Apple II systems. Merritt touched upon the announcement of the lower cost Macintosh systems in response to perceived markets, and explained that part of the function of the Russo organization is to insure that Apple II alternatives are not overlooked. The intention is that Apple should be concerned with user satisfaction with their products regardless of the platform used.

It's too early yet to perceive what affect the Russo organization will have on Apple's real-life actions. But the presence of representatives at the regional Apple II conferences indicates that Apple has heard the complaints about its past policies, and is now looking to gather reasonable suggestions for improving things. When Jim Merritt gave his closing speech, his notebook was close to full.

One very strong positive response to criticism is demonstrated by Apple's announcement of changes (effective June 25th, 1990) to the Apple Partners and Associates developer support programs. With the new changes, Apple Associates will receive invitations to the annual Apple Worldwide Developer's Conference, and will also be eligible for a limited number of discounted hardware purchases to ease their development efforts. Formerly, the invitations and hardware discounts were limited to Apple Partners only.

In addition, a often-requested option of Apple II-only program pricing has been introduced; you can become an Apple II-only Apple Partner for $300 a year (Apple II and Macintosh price remains $600) and an Apple Associate for $150 (Apple II and Macintosh remains $350). This brings the cost of these programs back into range for many smaller Apple II developers, many of which had complained the original prices were disproportionately high given the dominance of Macintosh specific information supplied as part of the program. (For more information about the programs, you can call the Apple Developer Hotline at 408-974-4897.)—DJD

Apple acquires Claris

Possibly in line with some of it's stated aims, Apple has decided to acquire Claris Corporation as a wholly-owned subsidiary. Claris had existed as a majority-owned subsidiary of Apple since it was "spun off" by Apple in 1987 (see "Ask Uncle DOS", June 1987, p. 3.37); Apple has chosen to bypass the option of allowing Claris to become a separate publicly-owned company. The current CEO of Claris, William V. Campbell, will remain but now reports to John Sculley.

According to John Sculley, Claris will focus on software development and marketing in three areas: continuation of current "flagship applications" (is it absurd to assume the dominating integrated applications AppleWorks and *AppleWorks GS* will be part of this group?), extensions to Apple's system software to enrich the Apple software development environment and "promote the creation of innovative Macintosh applications", and products to "bridge Macintosh with other computing environments". Consideration will be given to licensing technologies Claris may develop to software developers for incorporation in new applications.

It's interesting to consider why Apple may have made this move. Part of the original justification for Claris's formation was to reduce any perception of a "head to head" competition between Apple and third party application developers (Apple, of course, retains proprietary control of its system software); retaining ownership of Claris seems to negate that intent of the "spin-off". There is the possibility that Apple feared an independent Claris might start developing versions of its major applications for Microsoft's graphics-based operating system for IBM PC compatibles, *Windows 3.0*. Is Apple positioning to reduce its dependence on Microsoft as a source of application software for the Mac, or is Apple trying to "rein in" Claris, or was Apple fearful of the possible acquisition of a "public" Claris by a competitor?

Again demonstrating mixed signals on its commitment to the Apple II, Apple's press release regarding Claris gave a good example of the work the Russo organization has cut out for it. In the three items delineating Claris's part in Apple's strategy, only the Macintosh was explicitly mentioned by name (twice). It is just such omissions (even assuming it was an oversight) that allows the persistence of speculation regarding the *future* of the Apple II, because the Apple II is not *overtly* mentioned as is repeatedly allowed for the Mac.

It's probably going to be the job of the Russo organization to point out such inequities to Apple's top management. If Apple's job is to sell computers (and it must be assumed that *is* what it's all about), then they should insure *all* their literature reflects optimum promotion of *both* Apple II and Mac platforms.

Apple claims to be interested in the Apple II customer. Making sure the Apple II is mentioned in comments involving *future* strategy would be a *simple* place to start demonstrating commitment to that interest. That has been accomplished verbally in recent weeks, but Apple's message still comes across differently in print.—DJD

Object Module Manager

The inherent problem with 8-bit software on the Apple II has always been the lack of resource management. To be fair, this was a limitation in all of the popular 8-bit personal computers, but only the Apple II has survived long enough for the problem to become painfully obvious versus its present-day competitors. (GS/OS applications using the ToolBox on the IIgs do not have this problem; they have a full-featured Memory Manager and other protocols available to arbitrate things.)

The problem is most apparent in Applesoft, where short machine-language programs are often used to enhance the language. The eternal question is where to put such routines in the Apple II memory so that they stay safe.

Roger Wagner Publishing's *ToolBox* series for Applesoft takes one approach: it attaches the routines to the parent Applesoft program itself so that whenever the program is loaded, the routines come with it. But what about a program where it is more desirable to have a series of installed enhancements that can be accessed by several modular "subprograms"?

Tom Weishaar (with some help from Mark Simonsen of Beagle Bros) described a protocol of adding routines to ProDOS BASIC using the EXTRNCMD vector at $BE06 in the BASIC.System global page. EXTRNCMD can be altered to point to a user-installed external command which executes as if it were a BASIC.System command. Tom used this to implement a ProDOS "TYPE" command that types the contents of a file (usually a text file) as displayable ASCII characters; the program was presented in a *Softalk* magazine article and later incorporated as Chapter 20 of *ProDOS Inside and Out*. (Incidentally, the listing in the book ended up missing a couple of lines thanks to yours truly; you can write me at ***A2-Central*** to request the corrections before you try entering the program.)

The larger problem the protocol did not solve was the management of multiple routines in memory. Removing a routine (other than the last one added) and recovering the memory it used was not feasible without a more complicated "master routine" that could recognize when it was "safe" to add or remove a routine, and then identify and relocate the remaining routines. The Morgan Davis Group has now provided such a utility, the *Object Module Manager* ($24.95 from The Morgan Davis Group, 10079 Nuerto Lane, Rancho San Diego, Calif. 92078-1736, 619-670-0563 or FAX 619-670-9643).

The centerpiece of the *Object Module Manager* is the Loader, which manages loading the modules from disk (relocating memory resident modules to close any gaps in memory, if need be), communication between the modules, and purging of modules no longer needed. The Loader is installed by executing the system file OMM.LOADER; if the Loader is already installed, a duplicate installation will be aborted with a "DUPLICATE FILE" error.

The Loader adds seven commands to BASIC through Applesoft's ampersand command "hook". Each command is of the form "& <command> <list>" where "<command>" is the actual *Object Module Manager* command name and "<list>" is data that can be passed to the command.

The LOAD GET command accepts one or a series (separated by commas) of module names and loads them from disk:

```
& LOAD GET "module1"
```

would load the module named "module1" from disk, and

```
& LOAD GET "module1","module2"
```

would load "module1" and "module2". The Loader includes a "trace" mode (enabled as a default) that will display information regarding each module file as it is loaded: an index number, an ID number, default address (for relocation calculations), length, date, title, and version. The ID numbers for each module must be distinct; attempting to load a module with the same ID as a previously loaded module will abort with a "DUPLICATE FILE" error.)

LOAD FRE accepts a module index number or ID number and removes the associated module from memory, relocating other modules to fill any gaps in memory use. With trace mode on, a report on the "freed" module will be displayed as it is purged. The index number is distinguished as a value that will always be 255 (the maximum number of memory resident modules allowed) or less, while the ID will always be 256 or greater. The *Object Module Manager* Loader itself will always have an index number of 0, and

```
& LOAD FRE 0
```

will therefore unload the Loader and all modules.

LOAD TRACE and LOAD NOTRACE enable and disable the "trace" mode respectively. LOAD PRINT prints the "trace" information for all installed modules. LOAD PEEK allows a program to determine information about loaded modules:

```
& LOAD PEEK 0,N
```

will return the count of available modules into the numeric variable N, and

```
& LOAD PEEK 1,M,A$
```

returns the "trace" information for the module with the ID (or index) number contained in the variable "M" into the string variable A$.

The interface to each module's commands is through the LOAD CALL command. LOAD CALL requires three pieces of data: the ID (or index) number for the desired module, a message code, and a function number. There also may be extra data items to pass to the called function; these can be added after the three required items.

Object Module Manager requires that the modules it uses be *relocatable*; that is, the modules must be constructed in such a fashion as to allow the OMM.LOADER to move them about in memory. To this end, each module must be of type REL (relocatable object file) and designed with a specific header for recognition and manipulation by the Loader. To allow OMM.LOADER's relocator to operate, any object code that might require modification during relocation must be organized and identified as a contiguous block, and any data identified as a separate block, rather than intermixing the two (the data block can contain 65xxx instructions as long as no absolute address references to points within the module are used).

Modules may send messages to each other; by this method a module can ask another module to execute a task needed for the originating module to complete a function. By using messages, the "sending" module does not have to know details about the internal structure of the "receiving" module. Each module must respect a minimum set of commands; the big three are MSG_INIT (initialize the module), MSG_INFO (return module information string), and MSG_QUIT (shut down module). If the module executes ampersand commands itself (reasonable to assume if the module is to have a function), a MSG_AMPR (execute ampersand service routine) command must also be supported. In addition, the *Object Module Manager* itself may send other messages: MSG_USER (intermodule service request), MSG_REL1 (alert module that it is about to be relocated), MSG_REL2 (alert module that it has just been relocated), MSG_KILL (notification that a module is about to be removed), MSG_DIED (notification that a module has been removed), MSG_BORN (notification that a module has just been installed), MSG_IDLE (idle event; not yet implemented). Most of these are designed to allow modules to recognize when management functions may affect them; interdependent modules will need these whereas other modules may be able to ignore them.

Several service routines are built into the *Object Module Manager* that a module may access: OMM_GETID returns the index of a module based on a supplied ID number, OMM_XOAMP executes the "original" ampersand vector (the "&" vector in effect before the Loader was installed), OMM_FREE frees a module, OMM_PUTWORD stores a word value to an Applesoft numeric variable, OMM_PUTSTR stores a string to an Applesoft string variable, OMM_GETSTR evaluates the string expression at the current position of the Applesoft text pointer and returns its descriptor, OMM_PADDEC prints a word in decimal (right justified), and OMM_C2PSTR copies a string into a Pascal format string (with a leading "length" byte).

The package comes with *ORCA/M* and *Merlin* interface files containing "equates" for the *Object Module Manager*. You can use the equates with any source code that you develop to create your own modules. The use of assembly allows you to code for whatever processor you have available (6502, 65C02, 65802, 65816, ...) but it will be your responsibility to make sure the proper environment is present.

Modules you create should allocate any buffer space you may need internally; the GETBUFR/FREBUFR mechanism of BASIC.System is not guaranteed to work between *Object Module Manager* service calls due to the dynamic management of the modules. The communication between modules also can't take place by absolute reference since the modules may be relocated when other modules are deleted.

There is a paucity of "standard" modules supplied; one sample module with source code is included along with documentation on disk. Apparently the product is aimed squarely at programmers who are interested in converting existing routines to take advantage of the *Object Module Manager* environment, or in developing new modules for distribution for use use with the environment. An *Object Module Manager* compatible adaptation of Morgan Davis's *AmperWorks* (minus the & FRE command) is included for purchasers of *ModemWorks* or *MD-BASIC* (*AmperWorks* is included with those products, but not sold separately) who wish to use the AmperWorks commands in the *Object Module Manager* environment. You will need to own one of those products for the documentation of *AmperWorks*. (Although *MD-BASIC* requires a IIgs, notice that *Object Module Manager* and *AmperWorks* do not.)

To distribute programs you create with *Object Module Manager*, you must obtain a license from the Morgan Davis Group. For commercial programs or a single-site license, the fee is a one-time payment of $100.—DJD

Miscellanea

"Can the Machine Maim the Message?" That's the title of an *Academic Computing* (Jan. 1990) article by Marcia Peoples Halio that has been generating some controversy lately.

In the June 29 issue of *Personal Computing*, a letter titled "Scared by 'McStudents'" from Joe Celko (pp. 15-16) refers to the article. According to the letter, freshman students at the University of Delaware (where Dr. Halio is the assistant director of the writing program of the English department) were allowed to choose between Macintosh and IBM PC lab sections for their composition classes. The results were that students using the PCs wrote at an average grade level of 12.1 on the Kincaid scale versus an average of 7.95 for the students using the Macs. Several other quotes indicate other weaknesses in the performance of the students using Macs versus those using the PCs, including simpler and less diverse sentence structure, more spelling errors, and "fluffier" topics (rock music versus capital punishment, for example).

Many Macintosh (and even IIgs "desktop") afficianados I have mentioned this to have gone absolutely ballistic, refering to numerous "studies" that demonstrate just the opposite. I am aware of one such independent study (contracted with Diagnostic Research, Inc., by Apple) entitled "Macintosh or MS-DOS: a synopsis of what MIS managers and other professionals in Fortune 1000 companies have to say" that came to the conclusions that (for the companies studied) learning the Macintosh system was 34% easier than MS-DOS, and that Mac users reported using one-and-a-half times the number of software packages that MS-DOS users reported. Higher productivity figures were also reported for Macintosh.

(Those who read Bruce Tognazzini's "The Human Interface" column might expect me to also use some of his figures; alas I cannot. Although his January 1989 column asserts his figures are based on "scientific conducted" studies, his refusal to discuss Apple's testing adequately to allow the basic scientific principle of *independent repetition and evaluation of the experimental method* forces me to consider his assertions with the same skepticism as I do "flying saucers")

Until I see the original article (an interlibrary loan is being facilitated as I write this), I tend to believe that comments from "Gloom and Doom: Computers and Composition" in the June 1990 *mini'app'les* Newsletter (The Minnesota Apple Computer User's Group, Inc., Box 796, Hopkins, Maine 55343) are insightful. The author, Emmeline Gragerford, looks at Dr. Halio's article and offers some theories about the "fluffy Mac phenomenon".

Several of the observations reflect what I think is the most important key to interpreting the results: *the results are influenced by the temperment of the student's judgement which resulted in their selection of a CPU.* Students who select the Mac primarily for its visual interface appeal or ease of use versus the "user hostile" image of MS-DOS are already indicating a preference for a certain work environment (and possibly work ethic?). That theory could be tested by forcing students from each group to work with the "other" computer and see if the differences correspond to the individual students rather than the CPUs (descriptions of the study imply that the same students continued to use the same CPU). If the difference is the students, it would explain the differences from corporate studies, where the environment is likely to sort out work habits in short order.

Another factor Gragerford discusses is the propensity to play with aspects of the Mac display's flexibility, such as in trying various combinations of fonts. This resembles the "Computer Perseveration Syndrome" (***A2-Central***, April 1989, p. 5.17) where completed projects are often modified endlessly because the computer makes it easy. In truth, *content* must have priority over (visual) *style* when writing, but it's easy to forget when you know there is a particulary expressive font in the machine somewhere.

It's certainly not fair to blame the Mac for the work habits of its users, the survey results seem to be interpreted as meaning the Mac is not as satisfactory a tool as the PC in some cases. (This interpretation applies to other programs which use a graphics-based interface such as applications for GS/OS on the IIgs, *Windows 3.0* for MS-DOS, the Commodore Amiga, and others.) It is more reasonable to blame the software tool in use; possibly writing tools used for composition courses should be confined to more basic functions (and a minimum of distracting features), or the students should be advised how to concentrate on the use of the program as a writing tool.

The "computer as writing tool" message was driven home to me in a panel at the 1990 Chicago WindyCon science fiction convention. Editors advised prospective writers to throw their fancy fonts and graphics in a trash can when writing, because Real Editors want plain, easy-to-read type when evaluating the content of a story. In many cases, such as teleplays, there is a specific form expected, and submissions that don't conform may go unread. The fancy stuff can be saved until later; that's why we still write the newsletter using the Apple II text screen, and treat typesetting as a separate project.

There have been more changes at the top levels of Apple. On June 28 Lotus Development Corporation announced that Donald P. Casey would become vice president of their spreadsheet division effective July 15. If you don't have your Apple (re-)organization chart handy, Casey occupied the position of vice president, Networking and Engineering; the link between Ralph Russo and John Sculley.

Effective July 1, Robert Puette is the new president of Apple's Apple USA division, responsible for the division's sales, marketing, and support functions and channels. Puette came to Apple after 24 years at Hewlett-Packard Company where he led Hewlett-Packard's start-up personal computer business. He arrives as an individual with proven management experience.

Cirtech has announced *Duet*, a coprocessor that will allow the IIgs to run Macintosh programs. The card fits into any standard slot on the IIgs and contains a 68020 microprocessor with up to 8 megabytes of RAM and a custom ROM. The system recognizes standard Apple peripherals using the 65816 for input/output processing. Cirtech claims the system will outperform a Mac IIcx.

A 68882 math coprocessor can be added as an option to a socket on the *Duet* card. The card comes standard with one megabyte of RAM, and Cirtech anticipates having the card ready to ship by December 1990. No price has been determined yet.

By the way, Cirtech has a new address: Cirtech (UK) Ltd., Monksford Stables, Newtown St. Boswell's, MELROSE TD6 0RU, Scotland, 0835-23898 or FAX 0835-22471.

***ShrinkIt* for the IIgs, a desktop-based version of the standard Apple II archiving program,** has been completed by Andy Nicholas and is in distribution via national and local information services. This

Ask (or tell) Uncle DOS

This is the "be humble and admit your mistakes" issue...

*Before I wrote the answer to **"Bad Driving?"** (June 1990, pp. 6.38-39), I had received three separate reports of problems with 5.25 drives within two weeks, and had seen an anomaly with the **AMR 3.5** drive (different behavior on the ROM 01 and ROM 03 IIgs) myself. However, after my solicitation in "Ask Uncle-DOS" in the last issue there have been only two further reports of problems with 5.25 drives, and neither follows the description given for the earlier glitches. At this point, I have to assume Apple's changes to the drive port don't represent a significant compatibility problem for 5.25 drives. I apologize for indicating Apple could have foreseen the few problems that did arise, since the incidents seem to be unusual.*

In "Wipe it clean" (p. 6.46), the BASIC examples used to check the /RAM volume for files are incorrect; not only will the BLOAD command gag on /RAM's shortened directory with an "END OF DATA" error, I got the pointer to the file count wrong.

The following corrected line will prevent the "END OF DATA" error by limiting the number of bytes read from the /RAM volume directory "file":

```
PRINT CHR$(4);"BLOAD /RAM,A$2000,TSYS,L$100"
```

and this is the corrected line to verify the number of files:

```
IF (PEEK(8192+37)) + 256 * PEEK(8192+38)) THEN
   PRINT "/RAM NOT EMPTY!"
```

One of the items overlooked in the answer to "Don't touch that memory" (p. 6.46) is that ProDOS Technical Note #26 does specifically mention that if /RAM is enabled, all space above $800 in auxiliary memory may be used by an application after /RAM is removed (and /RAM should be re-installed upon completion). If /RAM is not enabled, then memory above $800 can be used at the programmer's discretion, but the areas indicated as "reserved" in the Apple documentation must be left alone. Application programmers should realize that if their application finds /RAM is disconnected, someone may already have used some of the auxiliary memory; if they happen to overwrite a driver that is supposed to continue to reside in auxiliary memory for use by several applications, disaster may result. What is needed is some type of "globally respected" memory management protocol for auxiliary memory.

*In contradiction to the reply to "BASIC networking", there is a way to access the possible "illegal" filenames of an AppleShare volume from ProDOS 8...sort of. Matt Deatherage at Apple Developer Technical Support pointed me at the right section of the **AppleShare Programmer's Guide for the Apple II Family, Beta Draft** (available to APDA members for $50, part #A2G0051/A), which has supplanted the **AppleShare Programmer's Guide for the Apple IIgs**.*

When you are using ProDOS with AppleShare installed, many commands to interface with the AppleTalk protocols are made available through a new MLI command ($42). One

IIgs version will work with the GS/OS extended file structure when creating and extracting file archives in the Apple II NuFX format. The IIgs version of *ShrinkIt* will extract files from Binary II "envelopes" (even those "squeezed" with Floyd Zink's *Binary Library Utility*) as well as America Online's ACU format. It will also create and extract AppleSingle files.

An exciting new capability is the extraction of files from several non-Apple II formats such as *StuffIt* (Macintosh), *ZOO* and *ARC* (MS-DOS, Amiga, and Atari ST), and *compress* (unix). Users can employ this feature to extract and convert graphics with the aid of utilities such as *SHRConvert* ($15 from Jason Harper, 1480 Michelle Ct. #A, Colorado Springs, Colo. 80916); programmers will be able to use it to access public source code from foreign environments for conversion and use on the Apple II.

ShrinkIt for the IIgs requires 768K of memory and System 5.0.2 or greater. It's free, but if you find it useful, Andy Nicholas would probably appreciate a contribution. His address is 8415 Thornberry Drive East, Upper Marlboro, Md. 20772; electronic addresses are SHRINKIT for GEnie, ShrinkIt for America-Online, and 70771,2615 for CompuServe.

***Proterm v2.2* is available from a new company, InSync.** The new version has been updated from version 2.1 (which was distributed by Checkmate Technology) to include several minor changes, including modification of the YMODEM and ZMODEM protocols to allow their reliable use with the GEnie network, a fix for a file transfer "lock-up" problem when switching the program disk with a data disk, full use of the Apple IIgs Memory Manager for memory allocation on the IIgs, an expanded (from 28 to 36 systems) dialing list, enhancement of the file selection system to handle up to 250 files in a directory, 19200 baud support in the HST modem driver, addition of a "defaults" window for entering commonly used settings, selection between the logon "siren" or the standard Apple II bell signal (or silence), and a tune-up of the VT-100 emulation.

Owners of previous versions can receive a disk upgraded to version 2.2 by sending their original diskette plus $10 in cash or check to InSync Software, PO Box 22146, Phoenix, Ariz. 85028, 602-992-5515 (sales) or 602-992-1345 (technical support). Or for $35 you can receive the entire new package including manual, reference cards, and disks. InSync says they did not receive a complete list of registered users of *Proterm*, so they're asking your help in spreading the word about the upgrade.

Optical character recognition (OCR) is coming to the Apple IIe and IIgs. Beagle Bros programmers Alan Bird and Rob Renstrom have announced their OCR software, *INWORDS*, scheduled for release in September 1990 through a new company, WestCode Software.

OCR software allows the use of scanning hardware to process characters on a document and convert them into a text format for use in programs that deal with textual information. This allows, for example, a printed document to be scanned and processed to produce a word processor file containing the text portion of the document.

INWORDS will scan up to 3000 characters per minute and will establish the feasibility of OCR for both the IIe and IIgs platforms. John Pothier of Vitesse passed the information to us and revealed that the software had been tested on Vitesse's Quickie scanner, but that the support of other scanning systems was open to WestCode. For more information contact WestCode Software, 11835 Carmel Mountain Road, Suite 1304-311, San Diego, Calif. 92128, 619-679-9200.

Apple's rework program for the ImageWriter LQ ends October 31, 1990. If you have bad line registration or excessive noise, you need to see your dealer before that expiration date to get the prescribed Apple remedies free of charge.

APDA has announced several new Apple II products. For those interested in developing software using the new capabilities of the DMA SCSI card, the *Apple II High-Speed SCSI Card Utilities* ($30; part #A0242LL/A) are now available. Disk #1 contains utilities that ship with the card; this disk can be licensed through

```
Apple Computer Software Licensing
20525 Mariani Avenue M/S 38-I
Cupertino, Calif. 95014
```

Disk #2 contains beta versions of device drivers for the Apple Scanner and Apple Tape Backup 40SC (for development of applications using those devices).

In addition, final versions of the *Apple IIgs Tools and Interfaces* are available; these include the 5.0.2 versions of the interface files for APW C and the APW assembler as well as updated utilities for dealing resources. The full package is $50, #A0240L/A; the update from the previous release is $25, #A0241L/A.

Also available is the Addison-Wesley edition of the *Apple IIgs Toolbox Reference Volume 3*, (#A0299LL/B; $39.95) which can also be ordered from Addison-Wesley or <ahem> us.—DJD

---

of the extensions is called the ProDOS Filing Interface (PFI), and it includes an FINaming call to allow you to tell ProDOS to accept the extended naming conventions of AppleShare volumes. There was one exception versus use of the GS/OS AppleShare FST: I couldn't get the use of the ":" (versus "/") separator to work for pathnames (I kept getting a $53 error from the MLI telling me the pathname was invalid). Here's a sample using the call to set the prefix to "/SV/Users/<Any User>" (the user folder on our server) and to delete the file "Illegal Name" from within that folder. Hang on; it's a biggie.

The assembly language program consists of four subroutines: SetLong, which gets the current "long filename" status and saves it, then enables the use of the extended pathname syntax; FixLong, which restores the initial status; SetPfx, which uses the ProDOS MLI to set a prefix; and Destroy, which uses the MLI to delete the named file. Assuming you have the ***ProDOS 8 Technical Reference*** *(and you should have it and the* ***AppleShare Programmer's Guide*** *if you are going to use these commands) most of the source should be straightforward. First, we define a couple of constants:*

```
PfxBufr  GEQU  $0280      usually system path
MLI      GEQU  $BF00
         ORG   $300       location for code
```

Then we issue a few MLI AppleTalk commands using our parameter list for the FINaming call. To enable the long naming conventions, we first get and save the current naming protocol (so that we can restore it when we finish) and then set the long naming conventions:

```
SetLong  START              call this to enable
         USING  FINaming     long naming
         LDA    #$00         used to request
         STA    DirectF      current settings
         JSR    MLI
         DC     H'42'
         DC     A2'FINaming'
         STA    ErrorRtn     check error status
         BCS    ErrExit
         LDA    NamConvF     save current
         STA    NamConvS     settings
         LDA    #$C0         enable long
         STA    DirectF      conventions
         STA    NamConvF
         JSR    MLI
         DC     H'42'
         DC     A2'FINaming'
         STA    ErrorRtn
ErrExit  CLC
         RTS
         END
```

We call FixLong when we exit to restore the original settings for the long naming convention:

```
FixLong  START
         USING  FINaming
         LDA    NamConvS     restore saved
         STA    NamConvF      configuration
         LDA    #$C0         enable mask
         STA    DirectF
         JSR    MLI          restore settings
         DC     H'42'
         DC     A2'FINaming'
         STA    ErrorRtn
         CLC
         RTS
         END
```

Next we have our data area for the various values. The AppleTalk parameter list consists of a byte identifying our call as asynchronous, the value $33 to identify the call request as FINaming, a word (two bytes) to hold any AppleTalk error returned (an AppleTalk error will be reflected in the error returned for the MLI call; our program only checks for the presence of an MLI error), and two bytes specific to the FINaming call. One is the Naming Convention Flag (NamConvF); bit 7 of this flag tells

ProDOS whether you want the ProDOS Device Table disabled (if 0, the table is enabled; if 1, the Device Table will not be updated as network volumes are mounted or unmounted), and bit 6 specifies whether the AppleTalk Filing Protocol Long Naming Conventions are used (Long Naming used if 1; ProDOS naming used if 0). The other is the Directional Flag (DirectF); bits 6 and 7 of this byte determine whether you set the Naming Convention Flag items as specified from your parameter list, or whether you obtain the current settings (return current mode if 0; set new mode if 1). Following these bytes, we have two bytes for storing the FINaming protocols found, and for any MLI error returned:

```
FINaming    DATA                AppleTalk parm list
            DC      H'00'       async byte
            DC      H'33'       FINaming command
            DC      I2'$0000'   AppleTalk result
DirectF     DC      H'00'       Directional flag
NamConvF    DC      H'00'       Naming convention
NamConvS    DC      H'00'
ErrorRtn    DC      H'00'
            END
```

Finally, we have our "normal" MLI calls that we need to call from Applesoft in order to deal with the "long" names. SetPfx will set the ProDOS 8 system prefix to the pathname stored at $280:

```
SetPfx      START
            USING   MLIParms
            JSR     MLI
            DC      H'C6'       set_prefix call
            DC      A2'pPrefix' parameter pointer
            STA     MLIError
            CLC
            RTS
            END
```

Destroy will delete the specified file:

```
Destroy     START
            USING   MLIParms
            JSR     MLI
            DC      H'C1'       destroy call
            DC      A2'pPrefix' parameter pointer
            STA     MLIError
            CLC
            RTS
            END
```

Both MLI calls use a common parameter list:

```
MLIParms    DATA
pPrefix     DC      H'01'       parm count
            DC      A2'PfxBufr' prefix/path pointer
MLIError    DC      H'00'       for errors
            END
```

The following Applesoft program uses the assembly language routines (placed in the file P8LONG) to switch ProDOS to the long naming conventions, set the prefix to the user folder, delete the "Illegal Name" file, and then switch back to the original ProDOS settings. It's crude, but it should at least give you an idea how the delete can be done (of course, you do have to have AppleShare installed or an error will result). Notice that we POKE the pathnames into the ProDOS system path buffer at $280 (640 decimal) because BASIC.System's pathname parser would gag on them:

```
1000  REM == delete AppleShare file ==
1010 D$ =  CHR$ (4)
1020  PRINT D$;"BLOAD P8LONG": REM at $300
1030 SL = 768:FL = 809: REM SetLong, FixLong
1040 ER = 838: REM MLI error return
1050  REM — set MLI to use long naming —
1060  CALL SL: IF  PEEK (ER) THEN  PRINT "MLI
    Error: "; PEEK (ER): STOP
1070  GOTO 2010: REM >>> do your stuff in here <<<
1080  REM — set "long name" use back to original —
1090  CALL FL: IF  PEEK (ER) THEN  PRINT "MLI
    Error: "; PEEK (ER): STOP
1100  END
2000  REM — test set_prefix and destroy —
2010  GOSUB 10000: PRINT "Error return: ";E: STOP
2020  GOSUB 11000: PRINT "Error return: ";E: STOP
2030  GOTO 1090
9999 :
10000  REM — to set prefix with our code —
10010 SP = 839: REM set prefix entry
10015 EC = 864: REM MLI error return code
10020 PFX$ = "/SV/Users/<Any User>": REM our test
    prefix
10030  POKE 640, LEN (PFX$): REM set length
10035  REM next line puts prefix in buffer
10040  FOR I = 1 TO  LEN (PFX$): POKE 640 + I,
    ASC ( MID$ (PFX$,I,1)): NEXT I
10050  CALL SP:E =  PEEK (EC)
10060  RETURN
11000  REM — to destroy file with our code —
11010 DS = 850: REM destroy entry
11020 EC = 864: REM MLI error return code
11030 PFX$ = "/SV/Users/<Any User>/Illegal Name":
    REM our test file
11040  POKE 640, LEN (PFX$): REM set length
11050  REM next line puts prefix in buffer
11060  FOR I = 1 TO  LEN (PFX$): POKE 640 + I,
    ASC ( MID$ (PFX$,I,1)): NEXT I
11070  CALL DS:E =  PEEK (EC)
11080  RETURN
```

This particular program will print the error result and stop after the set_prefix (line 2010) and destroy (line 2020) operations. If the error return is "0", you can type "CONT" to tell Applesoft to continue with the next line. You'll want better error handling in a real-world program.

All this assumes you have access privileges to delete the file (and obviously there's no easy way to defeat the access privileges). And it's still a lot more work than using a GS/OS "delete" command which can handle the AppleShare FST's naming conventions.

In response to the request for computer travel cases ("Carry me away", May 1990, pp. 6.30.31), Orbit Systems, 914 Searcy Way, Bowling Green, Ky. 42103, 502-782-0600 or 800-541-5421 (sales) let us know that they manufacturer cases for the transit of computers and other electronics.—DJD

Remotely interested

I read your neat article on "multimedia" ("Picture This", July 1990); there are a couple of things I'd like to discuss.

First, about the *VidClip* product you mentioned: I had contacted these folks some time ago after reading an article about *VidClip* in *VideoMaker* magazine. I wrote to them and they kindly sent me their instruction manual. The manual revealed the programs only worked with Sony equipment capable of receiving CTRL-L hex codes. It did not apply to any other make of VHS VCR or camcorder.

Second, there is the subject I'm really interested in, which wasn't talked about in your article. (No gripe intended. I'm happy to see the subject discussed at all.) A multimedia area crying for help that I would like to see the IIgs aimed at is video tape post-production editing and assembly for home use. I am positive the IIgs can do this easily with some simple infrared transducing hardware, I'm certain there are a lot of IIgs people like me out there with VHS VCR's and camcorders who would buy whatever it took to give the IIgs the capability to control their camcorders and VCR's via infrared codes or directly through DIN plugs. This would give us the control capability of our VHS camcorders and VCR's to do the entire editing and re-assembly tasks on our home videos via the overlay card. It would also include the necessary genlock ability to add animation, titling, transitions, etc.

With this kind of capability you could even do a lot more than patching together home videos of little Willie. You could cover the entire educational and presentation gamut.

Would it be so impossible to have the IIgs control VHS VCR's in the same way you have described its ability to control laser discs? Computer control of VCRs is nothing new. Most of the more expensive VCR's are now coming out with editing capability. And of course there is the (sigh...) entire Amiga line.

Also, maybe you've heard of the Direct-ED machine, which is nothing but a dedicated computer in a small box using an infrared transducer which 'learns' a VCR's command codes and gives the capability to do post-production editing with graphics, titling, etc. The graphics aren't all that good and it's slow as you said about *VidClip*. But so what? It does the job. In fact, when the Direct-Ed Machine first came out there was talk that they would produce a package composed of the transducer and software for computers like the Apple II's to do the same thing. I think the IIgs with the Video Overlay Card is capable of doing it better. The only roadblock I see is the lack of the infrared hardware required to turn IIgs commands into tape VCR remote commands. If we had software and this, we'd be off and running. Why are these VCR infrared remote codes so hard to find? Is there anywhere a guy can find out what these mysterious VHS VCR Remote codes are?

Jack Schachtebeck
Roseville, Calif.

*As mentioned, the **SIA III** does have a "decode" mode that will work for SR-based devices. The trouble is that there are many remote control systems that aren't SR-compatible.*

*For a more generic remote control solution: Jerry Kindall (8-bit editor of **8/16** and employee at Quality Computers) tipped us about a "hacker project":*

*"Dan DeMaggio (another technician at Quality) and I have been working on hardware and software for interfacing infrared remotes with the IIgs. We've got the "digitizer" working, which allows you to "record" infrared signals into the IIgs and save them. The "transmitter" will allow you to "play" the signals back from the IIgs and control any infrared-controllable device you have lying around. With appropriate software (like a **HyperStudio** XCMD) this little piece of hardware could do some pretty astounding things.*

"We're also going to try to do some software which will allow the IIgs to recognize remote control signals. In other words, you could

program the IIgs to treat your remote as a hex keypad or as a macro keypad, or to just allow you to advance through a slide show using the remote.

"The gadget plugs into the game port and requires about $10 worth of parts, all readily available at Radio Shack. And while I said it's for the IIgs, it should also work on the II+ and IIe.

"I'm not sure when we'll be finished with this, as we're really working on it in our spare time, but we'll keep y'all posted."

Dennis Ulm added a comment to the discussion on GEnie with the address of the Direct-Ed manufacturer: Videonics, Inc., 1370 Dell Avenue, Campbell, Calif. 95008, 800-338-EDIT.—DJD

Speaking of portables

Do you think we could convince anyone at the big Apple to shove a IIgs into the Mac Portable case, slap in a 30 megabyte hard disk and a SCSI port and sell it real cheap? I'd buy it. Of course I would insist on SIMMs on the motherboard expandable to 4 megabytes RAM.

Michael A. Ament
Antioch, Ill.

Sounds good here. The problem is that using the Mac Portable's high-performance LCD screen and selling the result "real cheap" might be mutually exclusive. But there may be room under the Mac Portable's price tag.

We have also had one reader suggest that using a common case design for the IIgs and modular Mac systems (the IIcx and IIci) might help cut costs since it would eliminate producing two distinct cases.—DJD

Wipe it cleaner

I'm upgrading my Apple IIe and one of the components I'll be selling is a 10 megabyte *ProFile* hard drive. Before I sell it, how can I be absolutely certain I've removed all information from it? It would be a disservice to my clients, as well as potential legal malpractice, to inadvertently allow some clever hacker to expunge any of those old files. I seem to recall that a simple re-format or *ProSel*'s "wipe volume" destroys only the file allocation information, not the files themselves. For complete peace of mind, I need to be certain I've returned the *ProFile* to its virgin state before I sell it.

Charlotte Roberts
Lafayette, Calif.

Ah, yes, the "dark side" of data recovery...if you have software that performs a true low-level format of the hard disk, then that should erase all data.

If not, then you can use a simple program to overwrite all blocks on the disk. The following Applesoft program will accomplish this:

```
1000  REM == fill volume ==
1010 D$ =  CHR$ (4)
1020 P$ = "/HARD1/": REM hard disk volume
1030 SV = 1: REM subdirectory/volume
1040 PTH$ = P$ + "SV." +  STR$ (SV):SV$ = PTH$
1050  GOSUB 1190
1060  PRINT D$;"CREATE ";PTH$
1070 SD = 1: REM subdirectory
1080 PTH$ = SV$ + "/SD." +  STR$ (SD):SD$ = PTH$
1090  GOSUB 1190
1100  PRINT D$;"CREATE ";PTH$
1110 F = 1: REM file
1120 PTH$ = SD$ + "/FILE." +  STR$ (F)
1130  GOSUB 1190
1140  PRINT D$;"SAVE ";PTH$
1150  IF F < 100 THEN F = F + 1: GOTO 1120
1160  IF SD < 100 THEN SD = SD + 1: GOTO 1080
1170  IF SV < 25 THEN SV = SV + 1: GOTO 1040
1180  END
1190  REM — display progress —
1200  VTAB 1: HTAB 1: CALL  - 868: PRINT PTH$
1210  RETURN
```

To use the program, format your hard disk (to clean the volume directory information to reflect a "blank" disk) and run the program from ProDOS BASIC. Line 1020 should be changed to contain the volume name of your hard disk (enclosed in "/" characters). If you have a large hard disk with multiple partitions, you'll need to repeat this operation for each of the partitions containing "sensitive" data.

The rest of the program creates subdirectories on the fly (up to 25 in the volume directory and up to 100 "sub-subdirectories" in each of those) and then tries to fill each with copies of the BASIC program itself, filling the hard disk. If all goes well, the program should terminate with a "DISK FULL" error.—DJD

Debugging help

Is there any software which works like *Maxwell's Demon* to debug assembly language programs under ProDOS?

R. S. Teesdale
Grand Rapids, Mich.

Maxwell's Demon *is a DOS 3.3-based assembly language debugger that allows you to execute a machine-language program slowly (or even one step at a time) while observing the changes in the 6502 processor's registers and stack on a specialized display. Debuggers also usually include a method of viewing and modifying memory, and of setting "break points" to stop execution of a program (running with the debugger present) when a particular condition (usually a value of the program counter) is met.*

APDA sells the ***ProDOS Assembly Tools*** *(#A2Z2021) for $35 which include a ProDOS-based debugger called* ***BugByter****. It is limited to the 6502 (not 65C02) instruction set, but should be usable for most ProDOS 8 programs.*

For IIgs GS/OS applications, APDA sells ***GSBug and Debugging Tools v.4.0B1*** *(#A0037LL/A, beta; available only to APDA members) for $30. The* ***GSBug*** *disk includes an "init" file and application version of the GSBug debugger, as well as three CDA's:* ***Loader Dumper*** *(displays tool set information),* ***Memory Mangler*** *(displays memory usage information from the Memory Manager), and* ***Scrambler*** *(shuffles memory blocks to identify problems with memory access in your code). Note that* ***GSBug*** *is* ***not*** *usable to debug ProDOS 8 applications.—DJD*

Fixing *PaintWorks* fonts

A minor discovery that could help: some users may have found that the Calligraphy font (***A2-Central*** disk, February 1989), when used in *PaintWorks Gold*, has its delicate curves brutally cut on the right by the text cursor when you exit the text mode with Escape. The trick is: type your line with another font (a "straight" one) then go back to the "Choose Font" option without touching Escape and choose "Calligraphy". Your line will automatically shift to untruncated characters. Don't forget to leave one or two extra spaces at the end to take care of the last character.

Chris Marker
Paris, France

Portable power

I am designing a "glass cockpit" for a small homebuilt aircraft which will incorporate navigational inputs as well as engine sensor inputs. The display will show a moving-map graphic of the current position (latitude and longitude) including airports in the area, as well as bar-graph information showing engine performance. The prototype on the ground is just begun. I need your help in locating certain pieces to make this system work in the air.

I am in need of a power supply that accepts a 12 volt input and which outputs the standard voltages for the Apple IIgs: +12 volts at 1 A, -12 volts at 0.25 A, +5 volts at 4 A, and -5 volts at 0.25 A. This need not be an Apple II replacement (in size or power) as mounting and power requirements have yet to be decided and are still flexible. Because space and weight are at a premium in the aircraft a voltage inverter (12 volts DC in; 120 volts AC out) is not an option.

I will also need a small color video display that accepts analog video inputs and will run on +12 volts DC. Digital color inputs are a possibility, but I would have to use another slot card (Applied Engineering's IBM video option) which results in another level of complexity I prefer to avoid.

An added bonus of using the IIgs is the built-in synthesizer's wonderful sound capabilities. High-quality digitized voice messages can be played into the pilot's headset to inform him of critical parameter values— a powerful attention-getter in an environment (the traditional cockpit) dense with so much critical instrumentation.

Dave Carpenter
Cupertino, Calif.

This project sounds like it's over our head (Ouch!—sorry).

There are linear (analog) integrated circuit-modules that can provide regulated voltage available at your local Radio Shack or other electronics stores. These may work in your situation, but will require constructing your own supply. The monitor sounds like a tougher item to locate. Maybe a reader can provide some advice on a monitor or applicable power supply module.—DJD

Dictionary patch

In connection with the letter by Jean-Guy Mariage in your May 1990 issue, I send you a patch which allows one to use the extra keyboard characters in one's custom dictionary. The patch is for AppleWorks 3.0. I got it from Alan Bird at Beagle Bros and it works fine. Words containing "other characters" are not broken down while checking spelling and they are checked properly via the custom dictionary (but they won't be displayed as "suggested spellings"). The patch goes as follows:

```
BLOAD SEG.WP,A$300,B$8FCB,L5
POKE 768,64
POKE 772,126
BSAVE SEG.WP,A$300,B$8FCB,L5
```

Dr. S. S. Datye
Akureyri, Iceland

We pass this patch along for those who may be interested in trying it. In a quick test, we discovered the patched spelling checker will ignore the special characters preceding the first (USA) ASCII character code in a word.—DJD

A better switch

Do you know about *SWCP*? It's a $15 shareware program by Tim Grams that lets you switch slots "on the fly." It's great when you want to use a 5.25 drive only occasionally and have a card in the slot. It's super.

Merry Perry
Baltimore, Md.

***SWCP** (software control panel) is a ProDOS 8 program that can execute (and archive) IIgs Control Panel settings in preparation for running another program. The program seems to be primarily oriented for switching (and initializing) slots under ProDOS 8. That may be enough for most users. Under GS/OS, necessary native drivers may not be loaded since GS/OS loads the drivers for the devices it finds during the boot process. Device interfaces you "switch in" may therefore not have a necessary driver loaded.*

If you want to obtain a version directly from the author on disk, it is $20 from Tim Grams, P.O. Box 462283, Garland, Texas 75046-2283.—DJD

GS/OS fan

Having been a system engineer, I am greatly impressed with the IIgs system software— it is very complex, yet easy to use; it is fast (compare it to *Windows* on a 80286!); and it is virtually bug free! I would guess that version 5.0.2 could contain over 350,000 lines of code, all with no significant errors!

For those readers without mainframe operating system experience, I one time performed an installation for IBM OS PCP on a 64K (that's 65,536 total system memory bytes) S360-30. The system installation took over *six hours* continuous run time (tell that to anyone who complains about the GS/OS Installer).

OS for S367/370 was not error free. Release #15 included a book of *known* errors that was over 1 inch thick (the book of corrected errors was over 3 inches thick). And OS/MFT or OS/MVT was never a dependable system. In normal operation it crashed at least once each hour! It really did not become dependable until MVS arrived in the late 70's.

In GS/OS, we have an operating system that is at least as complex as IBM's OS, and yet it can be run by complete nincompoops. It does not crash into the monitor, and it does not trash disk files or directories. It is state of the art— reliable, and easy to use. The people at Apple all deserve a big thanks!

Ken Lessing
Fresno, Ohio

We're not in a position to judge mainframe operating systems; we'll accept your comments without trying to sort out the acronyms.

*Anyone interested in learning about working on a large software project should read Frederick Brooks' **Mythical Man Month.** It covers the development of the original disk operating system for the IBM 360 mainframe and explains everything that was done wrong. It wasn't the development team's fault; this was the first project of such complexity ever attempted; there was much to be learned.—DJD*

Lower cost IIgs?

I find it strange that anyone should be taken in by the story that Apple Computer cannot produce Apple II equipment at a price the market will bear. The Apple II used to have its own line of equipment.

What's left? The RGB monitor, the processor itself, and the 5.25 drive. That's all, folks. The Apple II Video Overlay Card sells for a song compared to the Mac video overlay card.

The story I heard was that Apple Computer can produce a low cost Mac for less than the IIgs and sell the Mac for greater profit. There is the rub; the Mac has a larger market price. All this proves is that Apple Computer has not used their best talent to reduce the production cost of the IIgs. They have let the IIgs production go sour on them.

And Apple's answer to not producing a lower cost IIgs is, of course...shove Macs, expensive Macs, at the schools. Pushing a Mac into the school systems is the same as selling an IBM clone. The decision is the same for the school.

The IIgs can be booted over a network. A Mac on a network must have its own boot device. Does Apple Computer point out this IIgs advantage? No.

The IIgs could handle 4 megabytes of memory from day 1. How many years did it take before the Mac got its first megabyte? Does Apple Computer talk about the IIgs advantage? No.

The Apple II line was successfully using hard drives before the Mac had a hard drive available. Does Apple Computer mention this in their advertising? No.

The most successful integrated desktop program in the world runs on an Apple II. Will the Mac ever have this success? No.

Sigh, when will we ever learn?

Ken Kashmarek
Eldridge, Iowa

Your letter actually encompasses two approaches, lowering the price of the IIgs, or raising it's perceived value. Most of us who love the Apple II probably feel it's the latter approach Apple has most obviously avoided. At Apple Fiesta, Barney Stone indicated that he believed that the Apple II would be a visible part of Apple's 1990 fall promotions.

Another question of interest is whether Apple included marketing expenses in the cost of its IIgs systems. Apple has been roundly criticized for the inefficacy of it's 1989 fall promotion for its lower-cost systems, and that weight lies heavily around the neck of the IIgs (the only Apple II system included in the promotion).

*We've found it strange to note the number of Macintosh articles regarding the price of Mac systems, including Steven Levy's article "Just Say Low" in the July 1990 **MacWorld**. It contained a couple of insightful comments from Apple's Brodie Keast: that despite Apple's experiment in cutting the price of the Mac Plus last winter (so that it could be offered near or below the $1000 mark) sales did not rise remarkably, and that there were significant differences in the manufacturing costs for a more powerful model (the SE/30). It makes us wonder how much a "low-cost Mac" will actually affect the market. —DJD*

Vote with your wallet

I feel we can be on the brink of another Apple II revival with the reports in ***A2-Central*** and John Sculley's letter in *The Apple IIgs Buyer's Guide*. This represents the most encouragement in years.

I also agree with Paul Statt's view that to make the Apple II a success, it is up to us to purchase new hardware and software and to extend those subscriptions to our favorite publications. If there's a market, then there will be products produced to meet this demand. So, put your money where your heart is.

Lawrence Perry
Asheboro, N.C.

Apple can help the Apple II market by more effective advertising to increase interest and confidence in Apple II systems. Nobody can buy a system that they don't know about. No one (except hard-core enthusiasts) will buy a system if the company doesn't make its support for the system obvious.

*If Apple keeps that end of the bargain, then the viability of the market depends on the Apple II consumers. Paul Statt made the point in his **InCider** editorial that ultimately we vote with our wallets.*

Incidentally, don't overlook Mac companies in this regard. Many Mac peripherals will work on the IIgs, and it doesn't hurt to scan a Mac magazine for hard disk prices (for example) and make a few calls.—DJD

A2-Central™

[redacted]

© Copyright 1990 by
A2-Central

Edited by:

Dennis Doms

with help from:

Tom Weishaar	**Sally Dwyer**	**Dean Esmay**
Joyce Hammond	**Jeff Neuer**	**Jay Jennings**
Tom Vanderpool	**Jean Weishaar**	

A2-Central,—titled ***Open-Apple*** through January, 1989—has been published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $28 for 1 year; $54 for 2 years; $78 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first four volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

The full text of each issue of ***A2-Central*** is available on 3.5 disks, along with a selection of the best new public domain and shareware files and programs, for $84 a year (newsletter and disk combined). Single disks are $10.

Please send all correspondence to:

A2-Central
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

ISSN 0885-4017

Printed in the U.S.A.

GEnie mail: A2-CENTRAL
Voice: 913-469-6502
Fax: 913-469-6507